**Data Cleaning Steps**

1. **~~Handle Missing Values~~**~~:~~
   * ~~For~~ **~~time-series data~~**~~, interpolate missing values using methods like:~~
     + **~~Linear Interpolation~~**~~: Suitable for continuous data (e.g., prices, GDP, CPI).~~
     + **~~Forward/Backward Fill~~**~~: Use the last valid observation (useful for consistent trends).~~
   * ~~Alternatively, replace missing values with:~~
     + **~~Rolling Averages~~** ~~(e.g., 7-day or 30-day mean).~~
     + ~~A~~ **~~global median or mean~~** ~~value for non-temporal columns.~~
2. **~~Remove Duplicates~~**~~:~~
   * ~~Check for duplicate rows (if applicable) and remove them, especially for date-specific entries.~~
3. **Outlier Detection and Handling**:
   * Use statistical methods (e.g., Z-score or IQR) to detect extreme outliers in numerical columns like "gold price" or "silver volume."
   * Replace outliers with:
     + Median or rolling statistics.
     + Use **winsorization** to cap extreme values.
4. **Correct Data Types**:
   * Ensure all numerical columns (e.g., prices, volumes) are floats.
   * Convert date columns to a **datetime format**.
5. **Align Timeframes**:
   * Ensure all data is aligned to the same time intervals (e.g., daily, weekly).
   * Resample data for consistency, using methods like:
     + **Upsampling**: Fill gaps using interpolation for higher frequency data.
     + **Downsampling**: Aggregate data (e.g., mean, sum) for lower frequency.

**Data Transformation Steps**

**1. Normalization and Scaling**

* **Why?** Neural networks are sensitive to the scale of the inputs, especially RNNs and Transformers.
* Apply **Min-Max Scaling** (range 0-1) or **Standard Scaling** (z-score normalization): z=x−mean(x)std(x)z = \frac{x - \text{mean}(x)}{\text{std}(x)}z=std(x)x−mean(x)​
* Normalize time-series data independently for each feature.

**2. Feature Engineering for Time-Series Models**

* **Lagged Features**:
  + Add lagged values of target variables (e.g., previous day’s gold price).
    - Example: Gold Pricet−1,Gold Pricet−7\text{Gold Price}\_{t-1}, \text{Gold Price}\_{t-7}Gold Pricet−1​,Gold Pricet−7​.
* **Rolling Features**:
  + Compute rolling statistics (mean, variance, etc.) over a window (e.g., 7-day, 30-day).
* **Seasonality Features**:
  + Extract features from the date column:
    - Month, quarter, day of the week (e.g., 1 for Monday, 7 for Sunday).
    - Binary flags for weekends/holidays.
* **High-Low Spreads**:
  + Create new features for spreads (e.g., Gold High−Gold Low\text{Gold High} - \text{Gold Low}Gold High−Gold Low).

**3. Log Transformations**

* Apply log transformations to features like prices and volumes to stabilize variance and handle skewed distributions: Log-Transformed Value=log⁡(x+1)\text{Log-Transformed Value} = \log(x + 1)Log-Transformed Value=log(x+1)

**4. Encoding Categorical Data**

* If any categorical data (e.g., months) exists, encode it using:
  + **One-Hot Encoding**: For cyclic or non-ordinal features like "day of the week."
  + **Cyclic Encoding**: For cyclical features like "month": xsin=sin⁡(2π⋅month12),xcos=cos⁡(2π⋅month12)x\_{\text{sin}} = \sin\left(\frac{2 \pi \cdot \text{month}}{12}\right), \quad x\_{\text{cos}} = \cos\left(\frac{2 \pi \cdot \text{month}}{12}\right)xsin​=sin(122π⋅month​),xcos​=cos(122π⋅month​)

**5. Data Splitting**

* Split data into **training**, **validation**, and **testing** sets based on time (e.g., first 80% for training, 10% for validation, and last 10% for testing).

**6. Sequence Preparation**

* **RNNs** and **Transformers** require sequential inputs:
  + Create fixed-length input sequences (sliding windows).
    - Example: Use the previous 30 days’ data to predict the next day’s price.
  + Use tools like TimeseriesGenerator in Keras or create custom sequences:

python

Copy code

for i in range(len(data) - sequence\_length):

X.append(data[i:i + sequence\_length])

y.append(data[i + sequence\_length])

* Pad sequences if necessary for consistent input lengths (useful for Transformers).

**7. Handling Stationarity**

* Many time-series models perform better on stationary data. Apply transformations to achieve stationarity:
  + **Differencing**: Subtract the previous value from the current value: xt′=xt−xt−1x\_t' = x\_t - x\_{t-1}xt′​=xt​−xt−1​

**8. Dimensionality Reduction**

* Use **Principal Component Analysis (PCA)** to reduce redundant features while retaining most of the variance.

**9. Data Augmentation (Optional)**

* Augment time-series data using techniques like:
  + Noise injection to simulate minor fluctuations.
  + Smoothing or resampling for robustness.